# Unidad 4: Aseguramiento de Calidad de Proceso y de Producto

## Aseguramiento de calidad de proceso y de producto (PPQA):

Diagrama

Descripción generada automáticamente

### ¿Qué es la calidad?

Todos los aspectos y características de un producto o servicio que se relacionan con su habilidad de alcanzar las necesidades manifiestas o implícitas.

La calidad es relativa a las personas, a su edad, a las circunstancias de trabajo, el tiempo, etc.

### Importancia de trabajar para y con calidad en el desarrollo de software:

* **Satisfacción del cliente:** Cumplir con las expectativas y necesidades de los usuarios es fundamental para mantener una buena percepción de la empresa. El software debe satisfacer los requisitos funcionales, de performance y características implícitas para lograr esto.
* **Competitividad en el mercado**: En un mercado globalizado, la calidad es un aspecto diferenciador. Las empresas que implementan estándares de calidad pueden competir mejor y mantener la fidelidad de sus clientes, lo que a su vez incrementa los beneficios.
* **Reducción de costos**: La calidad no solo previene errores, sino que reduce el retrabajo. Implementar procesos de calidad desde el principio minimiza los costos asociados a correcciones, ya que detecta fallos antes de que se conviertan en problemas más costosos.
* **Eficiencia en el desarrollo**: Una buena gestión de calidad optimiza el uso de recursos, ya que se evitan retrasos y sobrecostos al cumplir con fechas de entrega y presupuestos planificados.
* **Reputación y confianza**: Las empresas que producen software de alta calidad retienen a sus clientes y construyen una reputación sólida. La certificación de estándares de calidad refuerza esta confianza en el mercado.

### Ventajas de trabajar con calidad:

* **Prevención de fallos:** Los errores se identifican temprano en el proceso de desarrollo, lo que reduce la necesidad de corrección en fases posteriores, donde es más costoso.
* **Satisfacción de múltiples partes interesadas**: No solo el cliente final se beneficia, sino también el equipo de desarrollo, la gerencia y los otros interesados en el proyecto, ya que se alinean las expectativas y se reduce la frustración.
* **Mejora continua**: El enfoque en calidad promueve una cultura de mejora continua y aprendizaje, asegurando que los equipos estén capacitados y actualizados en las mejores prácticas del sector.
* **Mayor predictibilidad en los proyectos**: Cumplir con los plazos de entrega y presupuestos planificados es más sencillo cuando se aplican procesos de calidad bien definidos, lo que ayuda a evitar sorpresas.

En resumen, trabajar con calidad es una inversión a largo plazo que mejora la eficiencia, reduce costos y aumenta la satisfacción del cliente, aunque puede presentar desafíos iniciales en términos de costos y resistencia al cambio.

### Desventajas de trabajar con calidad:

* **Costos iniciales:** Implementar sistemas de gestión de calidad puede ser costoso al principio, ya que requiere inversión en herramientas, procesos y formación del personal.
* **Complejidad en la implementación**: Para algunas organizaciones, integrar estándares y procesos de calidad puede ser un desafío, ya que requiere cambios culturales y estructurales.
* **Tiempo de planificación**: El enfoque en la calidad requiere una planificación exhaustiva desde el inicio, lo que puede incrementar los tiempos en las primeras etapas del proyecto.
* **Resistencia al cambio**: En algunas empresas, los equipos pueden resistirse a adoptar nuevas prácticas de calidad, lo que puede dificultar la implementación si no hay apoyo adecuado desde la gerencia.

### ¿Qué cosas ocurren frecuentemente en los proyectos de desarrollo de software?

* Atrasos en las entregas.
* Costos excedidos.
* Falta cumplimiento de los compromisos.
* No están claros los requerimientos.
* El software no hace lo que tiene que hacer.
* Trabajo fuera de hora.
* Fenómeno del 90-90.
* ¿Dónde está ese componente?

### Situación de proyectos de software:

Gráfico

Descripción generada automáticamente

### Un software de calidad satisface:

* Las expectativas del cliente.
* Las expectativas del usuario.
* Las necesidades de la gerencia.
* Las necesidades del equipo de desarrollo y mantenimiento.
* Otros interesados…

### Principios de calidad:

* La calidad no se “inyecta” ni se compra, debe estar embebida.
* Es un esfuerzo de todos.
* Las personas son la clave para lograrlo.
  + Capacitación.
* Se necesita sponsor a nivel gerencial.
  + Pero se puede empezar por uno.
* Se debe liderar con el ejemplo.
* No se puede controlar lo que no se mide.
* Simplicidad, empezar por lo básico.
* El aseguramiento de la calidad debe planificarse.
* El aumento de las pruebas no aumenta la calidad.
* Debe ser razonable para mi negocio.

### ¿Calidad para quién?

Diagrama

Descripción generada automáticamente con confianza baja

A la calidad se la puede analizar desde distintas perspectivas o visiones:

* **Visión del usuario:** esto tiene que ver con las expectativas que tiene el usuario para con el producto, y si este las satisface. Esta es quizás la más complicada de establecer, porque está en la cabeza de los usuarios, lo cual es una razón más por la que el principio de comunicación constante es crucial, para ir validando en todo momento si estamos en el camino correcto. El agilismo trata de incluir la visión de calidad del usuario a través del rol de Product Owner, el cual es ocupado por una persona con capacidad de decisión del cliente.
* **Visión del producto:** esto se asocia al nivel de satisfacción de los requerimientos particulares de cada producto.
* **Visión del proceso (manufactura):** si el proceso de desarrollo utilizado es el correcto para el producto que se desea desarrollar, es decir, el proceso aporta valor al producto y no produce desperdicios.
* **Visión del valor:** encontrar un equilibrio en la relación costo-beneficio, para obtener siempre el mayor valor para el cliente posible y, obviamente generar ganancias con el desarrollo del software.
* **Visión Trascendental:** esta visión es un tanto utópica, ya que se asocia a objetivos difíciles de alcanzar. Por ejemplo 0 defectos en el producto, pero son aquellos objetivos que motivan a seguir en busca de la mejora continua.

### Gestión de la calidad:

Diagrama

Descripción generada automáticamente

Todo lo que este fuera de dicha coincidencia será desperdicio o insatisfacción.

La gestión de calidad en productos son acciones sistemáticas de las organizaciones para lograr calidad, las cuales requieren equilibrio entre:

* **Calidad programada:** los alcances del producto planificados.
* **Calidad realizada:** lo que realmente se ha desarrollado del producto.
* **Calidad necesaria:** mínimas características que el producto debe tener, para que este satisfaga los requerimientos.

En el equilibrio obtenemos las expectativas de calidad que esperamos del producto y todo lo que esté por fuera de esta es desperdicio o insatisfacción.

### Calidad en el software:

Diagrama

Descripción generada automáticamente

### ISO/IEC 25000: Modelos

* Calidad en uso (ISO 25010: 2011).
* Calidad de producto (ISO 25010: 2011).
* Calidad de datos (ISO 25012: 2008).

Interfaz de usuario gráfica

Descripción generada automáticamente

#### Modelo de calidad de producto:

Modelo de Barbacci/SEI:

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Calidad del software (MCCALL):

Diagrama

Descripción generada automáticamente

### Calidad y proceso de desarrollo:

* Lo que sirve a unos puede no servir a otros.
* El proceso es el único factor <<controlable>> al mejorar la calidad del software y su rendimiento como organización.

Diagrama

Descripción generada automáticamente

#### Definición de un proceso de software:

Proceso: la secuencia de pasos ejecutados para un propósito dado (IEEE).

Proceso de software: un conjunto de actividades, métodos, prácticas, y transformaciones que la gente usa para desarrollar o mantener software y sus productos asociados (SW-CMM).

Diagrama

Descripción generada automáticamente

##### ¿Cómo lo definimos?

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

##### ¿Cómo es un proceso para construir software?

Diagrama

Descripción generada automáticamente

### Aseguramiento de calidad de software:

“Lo que no está controlado no está hecho”.

Objetivos de SQA:

* Realizar controles apropiados del software y el proceso de desarrollo.
* Asegurar el cumplimiento de los estándares y procedimientos para el software y el proceso.
* Asegurar que los defectos en el producto, proceso o estándares son informados a la gerencia para que puedan ser solucionados.

##### Administración de la calidad del software:

* Concerniente con asegurar que se alcancen los niveles requeridos de calidad para el producto de software.
* Implica la definición de estándares y procesos de calidad apropiados y asegurar que los mismos sean respetados.
* Debería ayudar a desarrollar una “cultura de calidad” donde la calidad es vista como una responsabilidad de todos y cada uno.

##### Reporte del grupo de aseguramiento de calidad (GAC):

* No debería reportar al gerente de proyectos.
* No debería haber más de una posición entre la gerencia de primer nivel y el GAC.
* Cuando sea posible, el GAC debería reportar alguien realmente interesado en la calidad del software.

La administración de calidad debería estar separada de la administración de proyectos para asegurar independencia.

##### Actividades de la administración de calidad de software (actividades relacionadas con el aseguramiento de la calidad del software):

* Aseguramiento de calidad:
  + Establecer estándares y procedimientos organizacionales de calidad.
* Planificación de calidad:
  + Selecciona los procedimientos y estándares aplicables para un proyecto en particular y los modifica si fuera necesario.
* Control de calidad:
  + Asegura que los procedimientos y estándares son respectados por el equipo de desarrollo de software.

#### Funciones del aseguramiento de calidad de software:

* Prácticas de aseguramiento de calidad:
  + Desarrollo de herramientas adecuadas, técnicas, métodos y estándares que estén disponibles para realizar las revisiones de aseguramiento de calidad.
* Evaluación de la planificación del proyecto de software.
* Evaluación de requerimientos.
* Evaluación del proceso de diseño.
* Evaluación de las prácticas de programación.
* Evaluación del proceso de integración y prueba de software.
* Evaluación de los procesos de planificación y control de proyectos.
* Adaptación de los procedimientos de aseguramiento de calidad para cada proyecto.

##### La administración de calidad y el desarrollo de software:

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

### Procesos basados en calidad:

Diagrama

Descripción generada automáticamente

### Calidad de procesos en la práctica:

* Definir procesos estándares tales como:
  + Cómo deberían conducirse revisiones.
  + Cómo debería realizarse la administración de configuración, etc.
* Monitorear el proceso de desarrollo para asegurar que los estándares sean respetados.
* Reportar en el proceso a la administración de proyectos y al responsable del software.
* No use prácticas inapropiadas simplemente porque se han establecido los estándares.

### Estándares y aseguramiento de calidad:

* Los estándares son la clave para la administración de calidad efectiva.
* Puede ser estándares internacionales, nacionales, organizacionales o de proyecto.
* Estándares de producto definen las características que todos componentes deberían exhibir, ejemplo: estilos de programación común.
* Estándares de procesos definen cómo debería ser implementados los procesos de software.

### Planificación de calidad:

* Un plan de calidad define los productos de calidad deseados y como serán evaluados, y define los atributos de calidad más significativos.
* El plan de calidad debería definir el proceso de evaluación de la calidad.
* Define cuales estándares organizacionales deberían ser aplicados, como así si es necesario utilizar nuevos estándares.

### Control de calidad:

* Este implica el control del proceso de desarrollo para asegurar que se siguen los estándares y procedimientos.
* Existen dos enfoques para el control de calidad:
  + Revisiones de calidad;
  + Evaluaciones de software automáticas y mediciones.

#### Revisiones de calidad:

* Este es el principal método de validación de la calidad de un proceso o un producto.
* Un grupo examina parte de un proceso o producto y su documentación para encontrar potenciales problemas.
* Existen diferentes tipos de revisiones con diferentes objetivos:
  + Inspecciones para remoción de defectos (producto);
  + Revisiones para evaluación de progreso (producto y proceso);
  + Revisiones de calidad (producto y estándares).

### Mejora de procesos:

* Algunos modelos para la mejora de procesos son:
  + SPICE: Software Process Improvement Capability Evaluation.
  + IDEAL: Initialing, Diagnosing, Establishing, Acting, Leveraging.

#### Modelo IDEAL:

Diagrama

Descripción generada automáticamente con confianza baja

Nos da el contexto para crear un proyecto cuyo resultado va a ser un proceso definido. Es un modelo cíclico que mejora el proceso existente en una organización.

En el inicio empieza buscando un sponsor (apoyo en la organización (económico)). Esto es importante porque una mejora de proceso nunca es crítica, siempre hay algo más importante, entonces si no tengo un aval para ejecutar este tipo de proyectos, suelen no terminar exitosamente. Debemos entonces analizar dónde estamos y dónde queremos ir (se le llama análisis de brecha).

Se trata de un modelo de mejora de procesos que sirve como guía para inicializar, planificar e implementar acciones de mejora. Su nombre se debe a las 5 fases que lo componen:

* **Inicialización:** se reconocen las necesidades de cambio en la organización, razones para iniciar, determinar metas buscadas al proponer un cambio en el proceso. Se requiere que la organización apoye esta decisión de mejora (sponsoreo).
* **Diagnóstico:** Establecer la madurez actual de la organización y los riegos asociados al proceso de mejora (revisar estado actual y futuro de la org.).
* **Establecimiento:** Durante la fase se elabora un plan detallado con acciones específicas, entregables y responsabilidades para el programa de mejora basado en los resultados del diagnóstico y en los objetivos que se quieren alcanzar. Para elaborar el plan se parte de definir las prioridades para el esfuerzo de mejora.
* **Ejecutar/Acción:** Efectuar los cambios y reunir información para aprender de la mejora. Se implementan las acciones planeadas en un proyecto piloto (no en todos los procesos de la organización, ya que es una prueba). Si la solución es satisfactoria para la org, se implanta en la empresa.
* **Aprendizaje:** busca garantizar que el próximo ciclo sea más efectivo. Durante la misma se revisa toda la información recolectada en los pasos anteriores y se evalúan los logros y objetivos alcanzados para lograr implementar el cambio de manera más efectiva y eficiente en el futuro. Extrapolar la mejora al resto de proyectos en caso de que sea exitosa la ejecución o realizar correcciones en caso de que fracase el proyecto piloto.

#### Modelo SPICE (Software Process Improvement Capability Evolution):

Es un modelo creado para la mejora de procesos de software (ISO 15504). Es una adaptación para la evaluación de procesos de desarrollo software por niveles de madurez, dividido en 6 niveles.

Es un modelo dual, teórico. Tiene 2 partes:

* Modelo de Calidad y modelo de mejora.
* SPICE + IDEAL: son ambos modelos de mejora de proceso. Estos modelos de mejora se usan para crear Proyectos de Mejora para una organización. El resultado de este proyecto va a ser un proceso definido que se usará para hacer Proyectos.

Este modelo establece conjuntos predefinidos de procesos con objeto de definir un camino de mejora para una organización. En concreto, establece 6 niveles de madurez para clasificar a las organizaciones.

Aplicación

Descripción generada automáticamente con confianza baja

### Modelo de Calidad:

* CMMI (Capability Maturity Model Integration):
  + Es la evolución del SW\_CMM.
  + Lo emite el Software Engineering Institute (SEI).
  + La versión vigente es la 1.3 y fue liberada en noviembre del 2010.
  + ISACA compra el CMMI Institute, desde el 2020, la versión vigente es Versión 2.0 de ISACA.
  + Uno de los modelos más implementados en todo el mundo.
  + No es una norma, y no se “certifica”, sólo se evalúa a través de profesionales reconocidos por el SEI como Lead Appraisers.

#### CMMI: Constelaciones:

Diagrama

Descripción generada automáticamente

#### CMMI: Representaciones:

* Por etapas:
  + 5 niveles, de 1 a 5.
  + Definidos por un conjunto de áreas de proceso.
  + Mira la organización en su totalidad y evalúa su madurez para obtener los resultados que quiere obtener.
  + Niveles indican “Madurez Organizacional”.
  + Similar al SW-CMM.
  + Provee una única clasificación que facilita comparaciones entre organizaciones.
  + Provee una secuencia probada de mejoras.
  + El primer nivel acreditable es el 2, porque los niveles de madurez empiecen desde el 2.

Forma

Descripción generada automáticamente

* Continua:
  + 6 niveles, de 0 a 5.
  + Definidos por cada áreas de proceso.
  + Mira un proceso individualmente y ve la capacidad de cada proceso individual.
  + Niveles indican “Capacidad” de un área de proceso.
  + Similar al EIA/IS-731.
  + Permite comparaciones sobre la base de cada AP.
  + Permite elegir el orden de las mejoras.

Gráfico, Gráfico de barras

Descripción generada automáticamente

#### CMMI – Roles – Grupos:

Lo importante es que exista alguien responsable de cubrir las actividades de cada uno de los roles o grupos.

#### CMMI: Niveles de la representación por etapas:

1. Inicial:
   * Procesos impredecibles.
   * Pobremente controlado.
   * Reactivo.
2. Administrado:
   * Procesos caracterizados por proyectos.
   * Es a menudo reactiva.
3. Definido:
   * Procesos caracterizados por la organización.
   * Son proactivos.
4. Cualitativamente administrado:
   * Procesos medidos y controlados.
5. Optimizado:
   * Foco en la mejora de procesos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

A medida que se incrementa el nivel se aumenta la productividad y calidad y se reduce el riesgo.

#### Áreas de proceso por nivel par CMMI v1.3:

Texto

Descripción generada automáticamente con confianza media

* Optimizado:
  + Administración de performance organizacional.
  + Análisis causal y resolución.
* Cuantitativamente administrado:
  + Performance del proceso organizacional.
  + Gestión cuantitativa del proyecto.
* Definido:
  + Desarrollo de requerimientos.
  + Solución técnica.
  + Integración de producto.
  + Verificación.
  + Validación.
  + Foco en el proceso organizacional.
  + Definición del proceso organizacional.
  + Capacitación organizacional.
  + Administración integrada de proyectos.
  + Administración de riesgos.
  + Análisis y resolución de decisiones.
* Administrado:
  + Administración de requerimientos.
  + Planeamiento de proyectos.
  + Monitoreo y control de proyectos.
  + Administración de acuerdo con el proveedor. Opcional, obligatorio si se terceriza algo.
  + Medición y análisis.
  + Aseguramiento de calidad del proceso y del producto.
  + Administración de configuración.

Calendario

Descripción generada automáticamente con confianza media

Lo más difícil es llegar al nivel 2, porque se pasa de una organización inmadura (que no sabe trabajar con proceso definidos, no sabe organizarse, no planifica y no mide nada) a una organización madura.

### Puntos claves calidad en el software:

* El software puede analizarse desde varias perspectivas: como proceso, como producto… la calidad también.
* La calidad del software es difícil de medir.
* El software como proceso es el fundamento para mejorar la calidad.
* Trabajar con calidad es más barato que hacerlo sin calidad.
* La mejora de procesos exitoso requiere compromiso organizacional.
* Existen varios modelos disponibles para dar soporte a los esfuerzo de mejora.
* La mejora de procesos en el software ha demostrado retornos de inversión sustanciales.

### CMMI cara a cara con Ágil:

* “Nivel 1”:
  + Identificar el alcance del trabajo. De soporte.
  + Realizar el trabajo. De soporte.
* “Nivel 2”:
  + Política organizacional para plantear y ejecutar. Neutral.
  + Requerimientos, objetivos o planes. De soporte.
  + Recursos adecuados. De soporte.
  + Asignar responsabilidad y autoridad. De soporte.
  + Capacitar a las personas. De soporte.
  + Administración de configuración para productos de trabajo elegidos. De soporte.
  + Identificar y participar involucrados. Neutral.
  + Monitorear y controlar el plan y tomar acciones correctivas si es necesario. Neutral.
  + Objetivamente monitorear adherencia a los procesos y QA de productos y/o servicios. Desigual.
  + Revisar y resolver aspectos con el nivel de administración más alto. Neutral.
* “Nivel 3”:
  + Mantener un proceso definido. Neutral.
  + Medir la performance del proceso. Desigual.
* “Nivel 4”:
  + Establecer y mantener objetivos cuantitativos para el proceso. Desigual.
  + Estabilizar la performance para uno o más subprocesos para determinar su habilidad para alcanzar logros. Desigual.
* “Nivel 5”:
  + Asegurar mejora continua para dar soporte a los objetivos. De soporte.
  + Identificar y corregir causa raíz de los defectos. Neutral.

#### Hipótesis:

* Tolerancia de CMMI a Ágil:
  + Hay áreas de procesos que:
    - Hay soporte,
    - Otras neutrales.
    - Otras en conflicto.
  + Soporte a evaluación en un ambiente ágil.
* Tolerancia de agile a CMMI:
  + Está la puerta abierta?
  + Es posible?

#### Diferencias:

|  |  |  |
| --- | --- | --- |
|  | CMMI | Métodos ágiles |
| Valores esenciales | * Medir y mejorar el proceso.   Imagen que contiene Texto  Descripción generada automáticamente   * Caracteristicas de las personas.   + Disciplinados.   + Siguen reglas.   + Aversión al riesgo. * Comunicación.   + Organizacional.   + Macro. * Gestión de conocimiento.   + Activos de proceso. | * Respuestas a clientes. * Mínima sobrecarga. * Refinamiento de requerimientos.   + Metáforas.   + Casos de negocio. * Características de las personas.   + Comfortable.   + Creative.   + Risk takers. * Comunicación.   + Person to person.   + Micro. * Gestión de conocimiento.   + Personas. |
| Características | * Mejora organizacionalmente.   + Uniformidad.   + Nivelación. * Capacidad/madurez.   + Éxito por predictibilidad. * Cuerpo de conocimiento.   + Cruzando dimensiones.   + Estandarizado. * Reglas de atajo.   + Desalentadas. * Comités. * Confianza del cliente.   + En la infraestructura del proceso. * Cargando al frente.   + Mover a la derecha. * Alcance de la vista [involucrado, producto].   + Amplio.   + Inclusivo.   + Organizacional. * Nivel de discusión.   + Palabras.   + Definiciones.   + Duradero.   + Exhaustivo. | * Mejora en el proyecto.   + Tradición oral.   + Innovación. * Capacidad/madurez.   + Éxito por darse cuenta de oportunidades. * Cuerpo de conocimiento.   + Personal.   + Evolucionando.   + Temporal. * Reglas de atajo.   + Alentadas. * Individuos. * Confianza del cliente.   + SW funcionando, participantes. * Conducido por pruebas.   + Mover a la izquierda. * Alcance de la vista [involucrado, producto].   + Pequeño.   + Focalizado. * Nivel de discusión.   + Trabajo en mano. |
| Enfoque | * Descriptivo. * Cuantitativo.   + Número científicos y duros. * Universalidad. * Actividades. * Estratégico. * “¿cómo lo llamaremos?”. * Gestión de riesgos.   + Proactiva. | * Prescriptivo. * Cualitativo.   + Conocimiento táctico. * Situacional. * Producto. * Táctico. * “sólo hazlo!”. * Gestión de riesgos.   + Reactiva. |
| Foco | * Foco de negocio.   + Interna.   + Reglas. * Predictibilidad. * Estabilidad. | * Foco de negocio.   + Externo.   + Innovación. * Performance. * Velocidad. |

#### Similitudes:

* Meta: organizaciones de alto desempeño.
* Ambas planean.
* Ambas son CMMs (Consultant Money Makers).
* Ambas tienen reglas [reglas = requerimientos del proceso].
  + La violación tiene serias repercusiones.
  + “SEPG” (grupo de proceso de ingeniería de software) & “´política de proceso”.
* Ninguno es completo.
* No nuevas ideas.
  + Basadas en la experiencia.
* Ninguno es aplicable a “cualquier proyecto”.

## Auditorías de software:

“Evaluación independiente de los productos o procesos de software para asegurar el cumplimiento con estándares, lineamientos, especificaciones y procedimientos, basada en un criterio objetivo incluyendo documentación que especifique:

1. La forma o contenido de los productos a ser desarrollados.
2. El proceso por el cual los productos son desarrollados.
3. Cómo debería medirse el cumplimiento con estándares o lineamientos.”.

Auditar porque:

* Porque se da una opinión objetiva e independiente.
* Porque permite identificar áreas de insatisfacción potencial del cliente.
* Porque nos permite asegurar al cliente que estamos cumpliendo con nuestras expectativas.
* Porque permite identificar oportunidades de mejora.

### Beneficios de las auditorías de calidad de software:

* Evaluar el cumplimiento del proceso de desarrollo.
* Determinar la implementación efectiva de:
  + El proceso de desarrollo organizacional.
  + El proceso de desarrollo del proyecto.
  + Las actividades de soporte.
* Dar visibilidad a la gerencia sobre los proceso de trabajo.

Resultado: mejores productos conllevan a clientes satisfechos y crecimiento del negocio.

### Tipos de auditorías de calidad de software:

* **Auditoría de proyecto:**
  + Valida el cumplimiento del proceso de desarrollo.
  + Es responsable de ver si el proyecto se ejecutó con el proceso que se dijo que se iba a ejecutar.
* **Auditoría de configuración funcional:**
  + Valida que el producto cumpla con sus requerimientos.
* **Auditoría de configuración física:**
  + Valida que el ítem de configuración tal como está construido cumpla con la documentación técnica que lo describe.

#### Auditorías de proyecto:

* Las auditorías de proyecto se llevan a cabo de acuerdo a lo establecido en el PACS (Plan de Aseguramiento de Calidad de Software).
* El PACS debería indicar la persona responsable de realizar estas auditorías.
* Las inspecciones de software y las revisiones de la documentación de diseño y prueba deberían incluirse en esta auditoría.
* El objetivo de esta auditoría es verificar objetivamente la consistencia del producto a medida que evoluciona a lo largo del proceso de desarrollo, determinando que:
  + Las interfaces de hardware y software sean consistentes con los requerimientos de diseño en la ERS.
  + Los requerimientos funcionales de la ERS se validan en el Plan De Verificación y Validación de Software.
  + El diseño del producto, a medida que DDS evoluciona, satisface los requerimientos funcionales de la ERS.
  + El código es consistente con el DDS.

#### Auditoría de configuración funcional:

* La auditoría funcional compara el software que se ha construido (incluyendo sus formas ejecutables y su documentación disponible) con los requerimientos de software especificados en la ERS.
* El propósito de la auditoría funcional es asegurar que el código implementa sólo y completamente los requerimientos y las capacidades funcionales descriptos en la ERS.
* El responsable de QA debería validar si la matriz de rastreabilidad está actualizada.

#### Auditoría de configuración física:

* La auditoría física compara el código con la documentación de soporte.
* Su propósito es asegurar que la documentación que se entregará es consistente y describe correctamente al código desarrollado.
* El PACS debería indicar la persona responsable de realizar la auditoría física.
* El software podría entregarse sólo cuando se hayan arreglado las desviaciones encontradas.

### Roles:

* Gerente de SQA:
  + Prepara el plan de auditorías.
  + Calcula el costo de las auditorías.
  + Asigna los recursos.
  + Responsable de resolver las no conformidades.
* Auditor:
  + Acuerda la fecha de la auditoria.
  + Comunica el alcance de la auditoría.
  + Recolecta y analiza la evidencia objetiva que es relevante y suficiente para tomar conclusiones acerca del proyecto auditado.
  + Realiza la auditoría.
  + Prepara el reporte.
  + Realiza el seguimiento de los planes de acción acordados con el auditado.
* Auditado:
  + Acuerda la fecha de la auditoría.
  + Participa de la auditoría.
  + Proporciona evidencia al auditor.
  + Contesta al reporte de auditoría.
  + Propone el plan de acción para deficiencias citadas en el reporte.
  + Comunica el cumplimiento del plan de acción.

### Proceso de auditoría:

1. Preparación y planificación.

* El auditado solicita la auditoria y junto con el auditor definen la fecha, el alcance y objetivo, acordando los participantes de esta.
* Se deben definir métricas de auditoría, como por ejemplo: esfuerzo por auditoría, cantidad de desviaciones, duración de la auditoría, cantidad de desviaciones clasificadas por auditor de CMMI, etc.

1. Ejecución.

* El auditor escucha lo que la gente dice que hace y luego ve la documentación, pide evidencia (lo que debería estar haciéndose).

1. Análisis y reporte del resultado.

* El auditor realiza el reporte de resultados y el auditado analiza la respuesta, puede o no estar de acuerdo con algunas prácticas y se deja asentado el documento final. Se evalúan los resultados, se hace una reunión de cierre y se hace entrega del reporte final.

1. Seguimiento.

* Se analizan las desviaciones y prepara un plan de acción que se envía al auditor para que lo revise y apruebe. En función del acuerdo entre auditado y auditor, puede que el auditor haga un seguimiento de las desviaciones que encontró hasta que considere que han sido resueltas.

#### Planificación – responsabilidades:

Diagrama

Descripción generada automáticamente

#### Ejecución – reunión de apertura responsabilidades:

Diagrama

Descripción generada automáticamente

#### Ejecución propiamente dicha – responsabilidades:

Diagrama

Descripción generada automáticamente

#### Análisis y reporte del resultado:

Diagrama

Descripción generada automáticamente

Esta fase está compuesta por las siguientes actividades:

* Evaluación de los resultados.
* Reunión de cierre.
* Entrega del reporte final.

##### Análisis y reporte del resultado – evaluación de los resultados:

Diagrama

Descripción generada automáticamente

##### Análisis y reporte del resultado – reunión de cierre:

Diagrama

Descripción generada automáticamente

##### Análisis y reporte del resultado – entrega del reporte final:

Diagrama

Descripción generada automáticamente

#### Seguimiento:

Diagrama

Descripción generada automáticamente

### Checklist de auditoría:

* El contenido general del checklist sería:
  + Fecha de la auditoría.
  + Listado de auditados (identificando el rol).
  + Nombre del auditor.
  + Nombre del proyecto.
  + Fase actual del proyecto (si aplica).
  + Objetivo y alcance de la auditoría.
  + Lista de preguntas.

### Ejemplo de preguntas – planificación de proyectos:

* ¿Existe un plan de proyecto?
* ¿Está actualizado el plan de proyecto?
* ¿Existe un responsable para cada actividad?
* ¿Se han asignado recursos para las actividades de soporte?
* ¿Están disponibles los planes para todos los involucrados?

El auditor deberá asegurarse que:

* Los planes estén basados en los requerimientos.
* Las actividades planificadas se han llevado a cabo.
* Todos los involucrados se han comprometido con la última versión de los planes.
* Los cambios a los planes se hayan aprobado por todos los involucrados.
* La decisión de esos cambios se haya documentado oportunamente.
* Se han identificado y comunicado los riesgos del proyecto.

### Ejemplo de preguntas – fase de requerimientos:

* ¿Existe un documento de especificación de requerimientos?
* ¿Se han identificado unívocamente los requerimientos?
* ¿Están descriptos cada uno de los requerimientos?

El auditor deberá asegurarse que:

* Se han revisado y aprobado los requerimientos por parte de todos los involucrados.
* Si existen cambios a los requerimientos, los mismos han seguido el correspondiente proceso de cambios y se han revisado y actualizado los planes de proyecto.

### Rol del auditor durante la auditoría:

* Durante la ejecución de la autoría, el auditor debería:
  + Escuchar y no interrumpir.
  + Observar el lenguaje corporal del auditado.
  + Manejar el propio lenguaje corporal.
  + Tomar notas.
  + Preguntar.
  + Repita lo que ha escuchado para asegurarse que ha comprendido.

### Técnica de cuestionario:

* Comenzar con preguntas de final abierto (quién, cuándo, cómo, qué, dónde).
* Realizar preguntas cortas y puntuales.
* Finalizar con preguntas de final cerrado para clarificar conceptos.

### Reacciones comunes de los auditados:

* Es frecuente que el auditado:
  + Quiera impresionar al auditor.
  + Esté ansioso o tensionado.
  + Sienta como si estuviese siendo examinado.
  + Utilice la autoría para quejarse acerca de la empresa.
  + Brinde demasiada información, diciendo cosas que el auditor no debería saber.
  + Esté enojado o nervioso.

### Técnicas y herramientas:

* Checklist.
* Muestreo.
* Revisión de registros.
* Herramientas automatizadas.

### Reporte de auditoría:

* Los contenidos básicos de un reporte de auditoría serían:
  + Identificación de la auditoría.
  + Fecha de la auditoría.
  + Auditor.
  + Auditados.
  + Nombre del proyecto auditado.
  + Fase actual del proyecto.
  + Lista de resultados.
  + Comentarios.
  + Solicitud de planes de acción.

### Lista de resultados:

Existen tres tipos de resultados:

* **Buenas prácticas:** práctica procedimiento o instrucción que se ha desarrollado mucho mejor de lo esperado.
  + Es algo mejor a lo que se esperaba.
* **Deviaciones:** requieren un plan de acción por parte del auditado.
  + Cualquier cosa que no se hizo o cualquier cosa que no se hizo como el proceso dijo que había que hacerse.
  + Hay grados: “no adecuado” o “necesita mejora”.
* **Observaciones:** sobre condiciones que deberían mejorarse pero no requieren un plan de acción.
  + Son cosas que se advierten porque si bien no son desviaciones pueden llegar a generar un problema o son riesgosas y se destacan a consideración si el equipo quiere mejorarlo o no.

#### Buenas prácticas:

Se deben reservar para cuando el auditado:

* Ha establecido un sistema efectivo.
* Ha desarrollado un alto grado de conocimiento y cooperación interna.
* Ha adoptado una práctica superior a cualquier otra que se haya visto.

#### Desviaciones:

* Cualquier desviación que resulta en la disconformidad de un producto respecto de sus requerimientos.
* Falta de control para satisfacer los requerimientos.
* Cualquier desviación al proceso definido o a los requerimientos documentados que cause incertidumbre sobre la calidad del producto, las prácticas o las actividades.

#### Observaciones:

* Opinión acerca de una condición incumplida.
* Práctica que debe mejorarse.
* Condición que puede resultar en una futura desviación.

### Métricas de auditoría:

Cada organización deberá establecer las métricas más apropiadas. Algunos ejemplos serían:

* Esfuerzo por auditoría.
* Cantidad de desviaciones.
* Duración de auditoría.
* Cantidad de desviaciones clasificadas por PA de CMMI.

### Puntos clave de las auditorias:

* Las auditorías al proceso de desarrollo de software son tres:
  + Auditoría de proyecto.
  + Auditoría de configuración física.
  + Auditoría de configuración funcional.
* Las auditorías implican esfuerzo y costo para los proyectos, sin embargo sus beneficios son superiores.
* Son un instrumento para el aseguramiento de calidad en el software.

## Testing:

* Proceso destructivo de tratar de encontrar defectos (cuya presencia se asume) en el código.
* Se debe ir con una actitud negativa para demostrar que algo es incorrecto.
* Busca demostrar que el software hace lo que tiene que hacer y de la manera correcta.
* Es destructivo porque busca romper el software, busca romper lo que se está probando para encontrar los lugares donde falla.
* Testing exitoso: es el que encuentra defectos.
* Mundialmente: 30 a 50% del costo de un software confiable.

Gráfico, Diagrama

Descripción generada automáticamente

Está dentro de las actividades para el aseguramiento de calidad de software.

Auditoria de proyecto = si el proyecto está cumpliendo lo que dijo.

Auditoria de calidad = quien contrala a quien hace calidad para ver si hace calidad.

Revisiones técnicas = son entre pares. No son jerárquicas. No puede ser la gente de calidad ni el jefe. Son para mejorar algún artefacto.

El testing incluye las 2 V: verificación y validación.

* Verificación: ¿Estamos construyendo el sistema correctamente?
* Validación: ¿Estamos construyendo el sistema correcto?. Se hace foco en las pruebas de aceptación porque estamos validando con el cliente que el software cumpla con los requerimientos.

### Asegurar la calidad vs controlar la calidad:

Una vez definidos los requerimientos de calidad, debo tener en cuenta que:

* La calidad no puede “inyectarse” al final.
* La calidad del producto depende de tareas realizadas durante todo el proceso.
* Detectar errores en forma temprana ahorra esfuerzos, tiempo y recursos.
* La calidad no solamente abarca aspectos del producto sino también del proceso y como éstos se pueden mejorar, que a su vez evita defectos recurrentes.
* El testing NO puede asegurar ni calidad en el software ni software de calidad.

El aseguramiento de calidad implica detección temprana, tratar de prevenir. La prevención es más barata.

### Conceptos error vs defecto:

Error cuando se lo identifica antes de la etapa siguiente.

Como el testing es la última etapa ya estamos hablando de defectos.

Los errores de programación son corregibles, los errores de requerimientos no son corregibles. Algunos errores de arquitectura son corregibles.

No todos los errores y defectos provocan fallas. Las fallas son un mal funcionamiento de sistema.

El defecto es el que se encuentra en una etapa posterior a la que se inyecto, se encuentra en producción. Es más caro corregir un defecto que un error.

El error es el que se encuentra en una etapa anterior a enviarlo a producción.

Si en una prueba unitaria encuentro fallas es un error.

Las metodologías ágiles buscan, entre otras cosas, reducir el tiempo entre la definición de los requerimientos y las fases de desarrollo y pruebas. Al acortar este intervalo, cualquier falla detectada puede corregirse de manera más rápida y económica.

### Conceptos defectos, severidad y prioridad:

El testing exhaustivo (testear el 100%) es imposible. Para esto hay algunos criterios para priorizar.

Severidad: determina que tan grave es el defecto. La asigna la gente de testing.

1. Bloqueante. Es cuando el sistema no anda. Impide el funcionamiento del sistema.
2. Crítico. Provoca una falla, pero no una falla general, es una falla en alguna funcionalidad.
3. Mayor. Provoca una falla, pero nos permite seguir trabajando.
4. Menor. No impide la continuidad del sistema.
5. Cosmético. Es un error o defecto que no provoca falla (mal funcionamiento) y que normalmente tiene que ver con aspectos estéticos.

Prioridad: es del lado del negocio.

1. Urgente.
2. Alta.
3. Media.
4. Baja.

### Niveles de prueba:

Forma

Descripción generada automáticamente con confianza media

#### Testing unitario:

* Se prueba cada componente tras su realización/construcción.
* Solo se prueban componentes individuales.
* Cada componente es probado de forma independiente.
* Se produce con acceso al código bajo pruebas y con el apoyo del entorno de desarrollo, tales como un framework de pruebas unitarias o herramientas de depuración.
* Los errores se suelen reparar tan pronto como se encuentran, sin constancia oficial de los incidentes.

#### Testing de integración:

* Test orientado a verificar que las partes de un sistema que funcionan bien aisladamente, también lo hacen en conjunto.
* Cualquier estrategia de prueba de versión o de integración debe ser incremental, para lo que existen dos esquemas principales:
  + Integración de arriba hacia abajo (top-down).
  + Integración de abajo hacia arriba (bottom-up).
* Lo ideal es una combinación de ambos esquemas.
* Tener en cuenta que los módulos críticos deben ser probados lo más tempranamente posible.
* Los puntos clave del test de integración son simples:
  + Conectar de a poco las partes más complejas.
  + Minimizar la necesidad de programas auxiliares.

#### Testing de sistema:

* Es la prueba realizada cuando una aplicación está funcionando como un todo (prueba de la construcción final).
* Trata de determinar si el sistema en su globalidad opera satisfactoriamente (recuperación de fallas, seguridad y protección, stress, performance, etc.).
* El entorno de prueba debe corresponder al entorno de producción tanto como sea posible para reducir al mínimo el riesgo de incidentes debidos al ambiente específicamente y que no se encontraron en las pruebas.
* Deben investigar tanto requerimientos funcionales y no funcionales del sistema.

#### Testing de aceptación de usuario:

* Es la prueba realizada por el usuario para determinar si la aplicación se ajusta a sus necesidades.
* La meta en las pruebas de aceptación es el de establecer confianza en el sistema, las partes del sistema o las características específicas y no funcionales del sistema.
* Encontrar defectos no es el foco principal en las pruebas de aceptación.
* Comprende tanto la prueba realizada por el usuario en ambiente de laboratorio (pruebas alfa), como la prueba en ambientes de trabajo reales (pruebas beta).

### Ambientes para construcción del software:

* Desarrollo. Las pruebas unitarias se hacen acá.
* Prueba. Las pruebas de sistemas se hacen acá.
* Pre-producción. Las pruebas de aceptación se hacen acá, pero sino existe se hacen en producción.
* Producción.

Un ambiente es un espacio de trabajo. No se debería hacer pruebas en el ambiente de producción.

Se deben cumplir al menos: desarrollo, prueba y producción.

### Caso de prueba:

Es lo que diseña el analista de pruebas. Describe indicaciones paso a paso de como ejecutar el sistema en una funcionalidad en particular.

Un caso de prueba es la unidad de la actividad de la prueba.

Es una de las salidas de proceso de testing.

se define para ejecutar el mismo conjunto de pasos bajo las mismas condiciones tantas veces como sea necesario, con el fin de verificar si los defectos detectados han sido corregidos.

Consta de 3 partes:

1. Objetivo: la característica del sistema a comprobar.
2. Datos de entrada y de ambiente: datos a introducir al sistema que se encuentra en condiciones preestablecidas.
3. Comportamiento esperado: la salida o la acción esperada en el sistema de acuerdo a los requerimientos del sistema.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

* Set de condiciones o variables bajo las cuales un tester determinará si el software está funcionando correctamente o no.
* Buena definición de casos de prueba nos ayuda a reproducir defectos.
* “Los bugs se esconden en las esquinas y se congregan en los límites…”
* Objetivo: descubrir errores.
* Criterio: en forma completa.
* Restricción: con el mínimo de esfuerzo y tiempo.

### Derivación de casos de prueba:

Escala de tiempo

Descripción generada automáticamente con confianza media

### Conclusiones sobre la generación de casos:

* Ninguna técnica es completa.
* Las técnicas atacan distintos problemas.
* Lo mejor es combinar varias de estas técnicas para complementar las ventajas de cada una.
* Depende del código a testear.
* Sin requerimientos todo es mucho más difícil.
* Tener en cuenta la conjetura de defectos.
* Ser sistemático y documentar las suposiciones sobre el comportamiento o el modelo de fallas.

### Condiciones de prueba:

* Esta es la reacción esperada de un sistema frente a un estímulo particular, este estímulo está constituido por las distintas entradas.
* Una condición de prueba debe ser probada por al menos un caso de prueba.

### Estrategias de prueba:

Caja blanca y caja negra.

Se usan para determinar cuántos casos de prueba se van a necesitar para cada funcionalidad.

Buscan como hago para cubrir la mayoría de las cosas que tengo que probar.

Sirven para lograr la mayor cobertura de prueba (maximizar la cantidad de defectos encontrados) con el menor corto posible (en termino de plata, esfuerzo, etc.).

En el método de caja blanca se tiene acceso al código, mientras que en el método de caja negra se conocen los parámetros de entrada y las salidas esperadas.

#### Métodos:

* ¿Para qué usarlos? El tiempo y el presupuesto es limitado.
* Hay que pasar por la mayor cantidad de funcionalidades con la menor cantidad de pruebas.

#### Caja negra:

Las entradas y salidas están identificadas pero el proceso se desconoce.

Es con el sistema funcionando.

Son pruebas dinámicas, porque se ejecuta el software.

Hay 2 métodos de testing de caja negra:

* Basado en especificaciones.
  + Partición de equivalencias.
  + Análisis de valores límites.
  + Etc.
* Basados en la experiencia.
  + Adivinanza de defectos: ir a buscar el defecto porque se sabe que está ahí (por la experiencia se sabe).
  + Testing exploratorio: es sentarse a probar.

Basado en la experiencia: es basado en la experiencia de quien va a ejecutar el testing. No lo puede hacer cualquiera, porque lo tiene que hacer alguien que tenga experiencia.

##### Partición de equivalencias:

Imagen que contiene Logotipo

Descripción generada automáticamente

Una clase de equivalencia es un subconjunto de valores que cumplen con una condición (que se llama condición externa).

Se define una condición externa y de ahí, se definen las clases de equivalencias vinculadas con esa condición externa.

El concepto de equivalencia en el método de partición de equivalencia se refiere a que se asume que todos los valores dentro de una misma clase de equivalencia producirán un el resultado equivalente. Si funciona para uno de los valores, asumo que funciona para ese conjunto de valores.

Dos pasos:

1. Identificar las clases de equivalencia (válidas y no válidas).
   1. Rango de valores continuos.
   2. Valores discretos.
   3. Selección simple.
   4. Selección múltiple.
2. Identificar los casos de prueba.

##### Análisis de valores límites:

* Es una variante de la partición de equivalencias, en vez de seleccionar cualquier elemento como representativo de una clase de equivalencia, se seleccionan los bordes de una clase.

Utiliza solamente los valores extremos para probar.

No siempre va a existir valores límites, porque no siempre se trabaja con límites de fechas o numéricos.

##### Basados en experiencia:

* Adivinanza de defectos.
* Testing exploratorio.

#### Caja blanca:

* Se basan en el análisis de la estructura interna del software o un componente del software.
* Se puede garantizar el testing coverage.

Texto

Descripción generada automáticamente

Son pruebas estáticas, porque no se ejecuta el software para probar.

Se hacen en el código.

Se quiere ver la calidad del código, código malicioso, etc.

Clase de equivalencia = conjunto de casos de prueba -> condiciones externas.

Clase de equivalencias de entradas (cuales son los atributos que definen las entradas) y de salida (las salidas esperadas).

Solo hay una condición válida para una clase de equivalencia por ejemplo edad >= 18. Pero puede tener varias condiciones inválidas.

El debug y el testing de caja blanca no son lo mismo porque el debug es dinámico porque se debe ejecutar el software, además, el objetivo del debug es encontrar donde está el defecto (porque ya se sabe de la presencia del mismo); con el testing se busca si hay defectos, no se sabe si hay defectos.

### Ciclo de test:

Un ciclo de pruebas abarca la ejecución de la totalidad de los casos de prueba establecidos aplicados a una misma versión del sistema a probar.

El primer ciclo de prueba se llama ciclo 0.

Se debe tener idealmente 2 ciclos de pruebas (sino hay errores o defectos).

### Regresión:

Al concluir un ciclo de pruebas, y reemplazarse la versión del sistema sometido al mismo, debe realizarse una verificación total de la nueva versión, a fin de prevenir la introducción de nuevos defectos al intentar solucionar los defectos.

Es una estrategia de cómo se va a encarar el testing a partir del ciclo 1.

Sin regresión significa que solo se prueba de la lista de defectos.

Captura de pantalla de un celular

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Captura de pantalla de un celular con letras

Descripción generada automáticamente

### Proceso de pruebas:

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

#### Planificación y control:

* La planificación de las pruebas es la actividad de verificar que se entienden las metas y los objetivos del cliente, las partes interesadas (stakeholders), el proyecto, y los riesgos de la pruebas que se pretende abordar.
* Construcción del test plan:
  + Riesgos y objetivos del testing.
  + Estretegia de testing.
  + Recursos.
  + Criterio de aceptación.
* Controlar:
  + Revisar los resultados del testing.
  + Test converage y criterio de aceptación.
  + Tomar decisiones.

#### Identificación y especificación:

* Revisión de la base de pruebas.
* Verificación de las especificaciones para el software bajo pruebas.
* Evaluar la testeabilidad de los requerimientos y el sistema.
* Identificar los datos necesarios.
* Diseño y priorización de los casos de pruebas.
* Diseño del entorno de prueba.

#### Ejecución:

* Desarrollar y dar prioridad a nuestros casos de prueba.
* Crear los datos de prueba.
* Automatizar lo que sea necesario.
* Creación de conjuntos de pruebas de los casos de prueba para la ejecución de la prueba eficientemente.
* Implementar y verificar el ambiente.
* Ejecutar los casos de prueba.
* Registrar el resultado de la ejecución de pruebas y registrar la identidad y las versiones del software en las herramientas de pruebas.
* Comparar los resultados reales con los resultados esperados.

#### Evaluación y reporte:

* Evaluar los criterios de aceptación.
* Reporte de los resultados de las pruebas para los stakeholders.
* Recolección de la información de las actividades de prueba completadas para consolidad.
* Verificación de los entregables y que los defectos hayan sido corregidos.
* Evaluación de cómo resultaron las actividades de testing y se analizan las lecciones aprendidas.

Icono

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### El testing en el ciclo de vida del software:

Objetivos de involucrar las actividades de testing de manera temprana:

* Dar visibilidad de manera temprana al equipo, de cómo se va a probar el producto.
* Disminuir los costos de correcciones de defectos.

#### Modelo en V:

Diagrama

Descripción generada automáticamente

### Romper mitos:

* El testing es una etapa que comienza al terminar de codificar.
* El testing es probar que el software funciona.
* Testing = calidad de producto.
* Testing = calidad de proceso.
* El tester es el enemigo del programador.

### ¿Cuánto testing es suficiente?

* El testing exhaustivo es imposible.
* Decidir cuánto testing es suficiente depende de:
  + Evaluación del nivel de riesgo.
  + Costos asociados al proyecto.
* Usamos los riesgos para determinar:
  + Qué testear primero.
  + A qué dedicarle más esfuerzo de testing.
  + Qué no testear (por ahora).
* El criterio de aceptación es lo que comúnmente se usa para resolver el problema de determinar cuándo una determinada fase de testing ha sido completada.
* Puede ser definido en términos de:
  + Costos.
  + % de tests corridos sin fallas.
  + Fallas predichas aún permanencen en el software.
  + No hay defectos de una determinada severidad en el software.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

### Principios del testing:

* El testing muestra presencia de defecto.
* El testing exhaustivo es imposible.
* Testing temprano.
* Agrupamiento de defectos.
* Paradoja del presticida.
* El testing es dependiente del contexto.
* Falacida de la ausencia de errores.
* Un programador debería evitar probar su propio código.
* Una unidad de programación no debería probar sus propios desarrollos.
* Examinar el software para probar que no hace lo que se supone que debería hacer es la mitad de la batalla, la otra mitad es ver que hace lo que no se supone que debería hacer.
* No planificar el esfuerzo de testing sobre la suposición de que no se van a encontrar defectos.

### La psicología del testing:

* La búsqueda de fallas puede ser visto como una crítica al producto y/o su autor.
* La construcción del software requiere otra mentalidad a la de testear el software.

### Smoke test:

* Primera corrida de los test de sistema que provee cierto aseguramiento de que el software que está siendo probado no provoca una falla catastrófica.

### Tipos de pruebas:

* Testing funcional:
  + Las pruebas se basan en funciones y características (descripta en los documentos o entendidas por los testers) y su interoperabilidad con sistemas específicos.
    - Basado en requerimientos.
    - Basado en los procesos de negocio.
* Testing no funcional:
  + Es la prueba de “cómo” funciona el sistema.
  + No hay que olvidarlas. Los requerimientos no funcionales son tan importantes como los funcionales.
    - Performance testing.
    - Pruebas de carga.
    - Pruebas de stress.
    - Pruebas de usabilidad.
    - Pruebas de mantenimiento.
    - Pruebas de fiabilidad.
    - Pruebas de portabilidad.

### Pruebas de interfaces de usuarios:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

### Prueba de performance:

* Tiempo de respuesta.
* Concurrencia.

### Prueba de configuración:

Imagen que contiene Texto

Descripción generada automáticamente

### TDD:

“El acto de diseñar tests es uno de los mecanismos conocidos más efectivos para prevenir errores… El proceso mental que debe desarrollarse para crear test útiles puede descubrir y eliminar problemas en todas las etapas del desarrollo”.

* Desarrollo guiado por pruebas de software, o Test-Driver Development (TDD).
* Es una técnica avanzada que involucra otras dos prácticas: escribir las pruebas primero (Test First Development) y refactorización (Refactoring).
* Para escribir las pruebas generalmente se utilizan las pruebas unitarias.

## Revisiones técnicas:

Es una actividad realizada por un colega, cuyo propósito es mejorar la calidad de software, mediante la detección temprana de errores en cualquier artefacto que se genere, por ejemplo, en el código, requerimientos, diseño, arquitectura, riesgos, estimaciones, planes, etc.

Es un proceso estático de validación y verificación; y no corrige errores.

Objetivos:

* Introducir el concepto de verificación y validación.
* Presentar el proceso de verificación.

### Verificación y validación:

Diagrama

Descripción generada automáticamente

* Es un proceso de ciclo de vida completo.
* Inicia con las revisiones de los requerimientos y continúa con las revisiones del diseño, inspecciones del código hasta la prueba.
* Validación: ¿Estamos construyendo el producto correcto?
* Verificación: ¿Estamos construyendo el producto correctamente?
* **Falla:** error en un producto de trabajo.
* **Producto de trabajo:** salida de cualquier actividad correspondiente al ciclo de vida de desarrollo.
* Porqué existen las fallas:
  + Ruido de comunicación.
  + Limitaciones de memoria.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Debe planificarse:
  + Es un proceso caro.
  + Comienza en etapas tempranas.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Ejemplos:

* Fallas mayores:
  + En código:
    - Error lógico, estructural u otro que pueda ocasionar una falla operacional.
  + En diseño:
    - Una expresión en el diseño puede ocasionar una falla operacional si se implementara tal cual está especificado.
  + En requerimientos:
    - Una expresión en los requerimientos que pudiera ocasionar que no se cumpliera con las necesidades del cliente, o una expresión ambigua o información faltante que requerirá una investigación posterior.
  + En plan de prueba o casos de prueba:
    - Una condición que podría ocasionar que no se detectaran fallas en el programa o que la prueba no pueda llevarse a cabo o repetirse.
* Fallas menores:
  + En código o diseño:
    - Una violación a los estándares de codificación o de diseño (ej: comentarios en el código), que no ocasionará una falla operacional pero puede reducir la claridad y causar problemas de mantenimiento.
  + En requerimientos:
    - Un requerimiento que no pueda probarse.
  + En plan de pruebas o casos de prueba:
    - Información que no está clara o que pudiera causar que se requiera enfuerzo de testing innecesario debido a la redundancia.
* Notas cosméticas:
  + En documentación:
    - Errores de tipeo,
    - Errores ortográficos,
    - Errores gramaticales,
    - Se necesita actualizar el documento con una planificación más nueva (existe una versión más nueva).
    - Se necesita actualizar la historia de revisiones del documento.
  + En código:
    - Se necesita actualizar los datos de copyright de un código fuente utilizado.
    - Una sugerencia alternativa (ej: un algoritmo de búsqueda diferente).

Problemas del retrabajo:

* El retrabajo evitable corresponde al 40-50% del desarrollo.

Diagrama

Descripción generada automáticamente con confianza baja

Principios:

* La prevención es mejor que la cura.
* Evitar es más efectivo que eliminar.
* La retroalimentación enseña efectivamente.
* Priorizar lo rentable.
* Olvidarse de la perfección, no se puede conseguir.
* Enseñar a pescar, en lugar de dar el pescado.

Existen dos aproximaciones complementarias:

* Revisiones técnicas.
* Pruebas de software.

#### Revisiones técnicas:

* Proceso de V & V estático.
* Principal objetivo: detectar defectos y corregirlos en las etapas tempranas del desarrollo.
* Origen: 1976 Fagan introdujo proceso de inspecciones basados en experiencia en HW para detectar defectos lo más cerca posible de su generación.
* Practicadas por industria de SW en la que calidad y retrabajo son críticos.
* Muchas variantes respecto a las inspecciones de Fagan.
* Puede inspeccionarse cualquier representación legible del SW.
* Se aplican en varios momentos del desarrollo.
* El trabajo técnico necesita ser revisado por la misma razón que los lápices necesitan gomas: errar es humano.
* Algunas clases de errores se le pasan por alto más fácilmente al que los origina que a otras personas.
* Motiva a realizar un mejor trabajo.
* No requieren que el programa se ejecute.
* Ventajas:
  + Pueden descubrirse muchos errores.
  + Pueden inspeccionarse versiones incompletas.
  + Pueden considerarse otro atributo de calidad.
* Desventajas:
  + Es difícil introducir las inspecciones formales.
  + Sobrecargan al inicio los costos y conducen a un ahorro sólo después de que los equipos adquieran experiencia en su uso.
  + Requieren tiempo para organizarse y parecer ralentizar el proceso de desarrollo.
* Costos:
  + Infraestructura:
    - Entrenamiento.
    - Desarrollo/ajuste de plantillas e informes.
    - Desarrollo/ajuste de guías de lectura.
    - Implantación de programas de medición.
    - Herramientas de soporte.
  + Operacionales:
    - Tiempo individual y grupal.
    - Tiempo en completar informes.
  + Adicionales:
    - Preparar materiales, arreglar calendario, recolectar datos, etc.
    - Tiempo dedicado a la mejora de calidad.
* Tipos de revisiones:
  + Formales: tienen un proceso definido con roles.
    - Inspecciones (inspección de código de Fagan o inspección de Gilb).
  + Informales: cuando no existe un proceso de cómo realizarlo.
    - Walkthrough o recorrido.

|  |  |  |
| --- | --- | --- |
| Método | Objetivos típicos | Atributos típicos |
| Walktroughs | Mínima sobrecarga.  Capacitación de desarrolladores.  Rápido retorno. | Poca o ninguna preparación.  Proceso informal.  No hay mediciones.  No FTR. |
| Inspecciones | Detectar y remover todos los defectos eficiente y efectivamente. | Proceso formal.  Ckecklists.  Mediciones.  Fase de verificación. |

|  |  |
| --- | --- |
| Tipo de documento | Revisores sugeridos |
| Arquitectura o diseño de alto nivel | Arquitecto, analista de requerimientos, diseñador, líder de proyecto, testers. |
| Diseño detallado. | Diseñador, arquitecto, programadores, testers. |
| Planes de proyecto | Líder de proyecto, stakeholders, representante de ventas o marketing, líder técnico, representante del área de calidad. |
| Especificación de requerimientos | Analista de requerimientos, líder de proyecto, arquitecto, diseñador, testers, representante de ventas y/o marketing. |
| Código fuente | Programador, diseñador, testers, analista de requerimientos. |
| Plan de testing | Tester, programador, arquitecto, diseñador, representante del área de calidad, analista de requerimientos. |

|  |  |
| --- | --- |
| Métricas sugeridas | Fórmula |
| Densidad de defectos | Total de defectos encontrados / tamaño actual. |
| Total de defectos encontrados | Defectos mayor + defectos menor. |
| Esfuerzo de la inspección | Esfuerzo planning + esfuerzo preparación + esfuerzo reunión + esfuerzo retrabajo. |
| Esfuerzo por defecto | Esfuerzo inspección / total de defectos encontrados. |
| Porcentaje de reinspecciones | Cantidad reinspecciones / cantidad inspecciones. |
| Defectos corregidos sobre total de defectos | Esfuerzo inspección / tamaño actual |

##### Inspección:

* Es una actividad de garantía de calidad de software.
* Tiene un proceso formal y cuenta con un conjunto de roles.
* Es necesario la utilización de un checklist, que ayuda a la memoria para saber que cosas controlar.
* Se toman métricas y finalmente se realiza un reporte de la revisión al final de la inspección para analizar los defectos encontrados.
* Objetivos:
  + Descubrir errores.
  + Verificar que el software alcanza sus requisitos.
  + Garantizar que el software ha sido representado de acuerdo a cientos estándares.
  + Conseguir un software desarrollado de manera uniforme.
  + Hacer que los proyectos sean más manejables.
* Se lleva a cabo mediante una reunión y el éxito depende de su planificación.

|  |  |
| --- | --- |
| SON | NO SON |
| * La forma más barata y efectiva de encontrar fallas. * Una forma de proveer métricas al proyecto. * Una buena forma de proveer conocimiento cruzado. * Una buena forma de promover el trabajo en grupo. * Un método probado para mejorar la calidad del producto. | * Utilizadas para encontrar soluciones a las fallas. * Usadas para obtener la aprobación de un producto de trabajo. * Usadas para evaluar el desempeño de las personas. |

El proceso de inspección – roles participantes:

|  |  |
| --- | --- |
| Rol | Responsabilidad |
| Autor | * Creador o encargado de mantener el producto que va a ser inspeccionado. * Iniciar el proceso asignando un moderador y designa junto al moderador el resto de los roles. * Entrega el producto a ser inspeccionado al moderador. * Reporta el tiempo de retrabajo y el número total de defectos al moderador. |
| Moderador | * Planifica y lidera la revisión. * Trabaja junto al autor para seleccionar el resto de los roles. * Entrega el producto a inspeccionar a los inspectores con tiempo (48hs) antes de la reunión. * Coordina la reunión asegurándose que no hay conductas inapropiadas. * Hacer seguimiento de los defectos reportados. |
| Lector | * Lee el producto a ser inspeccionado. |
| Anotador | * Registra los hallazgos de la revisión. |
| Inspector | * Examina el producto antes de la reunión para encontrar defectos. * Registra sus tiempos de preparación. |

El proceso de inspección (convencional):

Texto

Descripción generada automáticamente

* Reunión de inspección:
  + Registro e informe de revisión decisión:
    - Aceptar el producto sin modificaciones.
    - Rechazo del producto.
    - Aceptar el producto provisoriamente.
  + Informe: ¿Qué se revisó? ¿Quién lo revisó?¿Qué se descubrió? Y conclusiones.
  + Duración de una reunión de inspección 2 horas máximo.

|  |  |  |
| --- | --- | --- |
| Operación | Código | Documentos |
| Planificación | 15 mins | 30 mins |
| Vista previa | 500 LOC/h | 500 líneas de texto/h |
| Preparación | 100 LOC/h | 140 líneas de texto/h |
| Inspección | 125 LOC/h | 140 líneas de texto/h |
| Mejora del proceso | 30 mins | 35 mins |
| Tamaño máximo por inspección | 250 LOC | 280 líneas de texto |

##### Walkthrough:

Técnica de análisis estático en la que un diseñador o programador dirige miembros del equipo de desarrollo y otras partes interesadas a través de un producto de software y los participantes formulan preguntas y realizan comentarios acerca de posibles errores, violación de estándares de desarrollo y otros problemas.

##### En resumen:

* Mejorar las pruebas:
  + Caro.
  + No sirve para remover errores en etapas tempranas.
* Recorridas -Walkthrought:
  + Buenos resultados, pero se toman pocas métricas.
  + No hay control del proceso.
* Inspecciones:
  + Mejores resultados, proceso controlado.
  + Métricas útiles a lo largo de todo el ciclo de vida del desarrollo.
* Puntos claves:
  + Revisar al producto, no al producto.
  + Fijar una agenda y cumplirla.
  + Limitar el debate y las impugnaciones.
  + Enunciar las áreas de problemas, pero no tratar de resolver cualquier problema que se manifieste.
  + Tomar notas escritas.
  + Limitar el número de participantes e insistir en la preparación por anticipado.
  + Desarrollar una lista de revisión.
  + Disponer de recursos y una agenda.
  + Entrenamiento.
  + Repasar revisiones anteriores.